**МИНОБРНАУКИ РОССИИ**

**Санкт-Петербургский государственный**

**электротехнический университет**

**«ЛЭТИ» им. В.И. Ульянова (Ленина)**

**Кафедра МО ЭВМ**

отчет

**по лабораторной работе №2**

**по дисциплине «Объектно-ориентированное программирование»**

Тема: Наследование

|  |  |  |
| --- | --- | --- |
| Студент гр. 6304 |  | Зыль С. Е. |
| Преподаватель |  | Терентьев А.О. |

Санкт-Петербург

2018

**Цель работы.**

Изучить наследование объектов в C++.

**Основные теоретические положения.**

*Наследование —* свойство системы, позволяющее описать новый класс на основе уже существующего с частично или полностью заимствующейся функциональностью. Класс, от которого производится наследование, называется базовым, родительским или суперклассом. Новый класс — потомком, наследником, дочерним или производным классом.

Каждый производный класс полностью сохраняет интерфейс родительского класса. Обратное, очевидно, неверно.

*Перегрузка методов* ***–*** возможность использования методов с одним именем, но с разными параметрами.

*Переопределение методов* – возможность языка программирования, позволяющая производному классу обеспечивать специфическую реализацию методов, уже определенных в базовом классе.

В C++ метод, обозначаемый ключевым словом “virtual”, может быть переопределен производным классом. Компилятор всегда сначала ищет возможность запустить метод, непосредственно объявленный в классе, и лишь при отсутствии такового будет искать метод в базовых классах.

**Постановка задачи.**

Необходимо спроектировать систему классов для моделирования геометрических фигур (в соответствии с полученным индивидуальным заданием). Задание предполагает использование виртуальных функций в иерархии наследования, проектирование и использование абстрактного базового класса. Разработанные классы должны быть наследниками абстрактного класса Shape, содержащего методы для перемещения в указанные координаты, поворота на заданный угол, масштабирования на заданный коэффициент, установки и получения цвета, а также оператор вывода в поток.

﻿﻿Необходимо также обеспечить однозначную идентификацию каждого объекта.

Решение должно содержать:

* условие задания;
* UML диаграмму разработанных классов;
* текстовое обоснование проектных решений;
* реализацию классов на языке С++.

**Ход работы.**

1. Написана структура Point, отвечающий за создание точек по координатам. Включает в себя данные (переменные x и y) и конструктор.
2. Написан класс «Shape», описывающий абстрактную фигуру. Класс хранит следующие переменные:

* Центральную точку типа Point
* Угол поворота фигуры типа int
* Цвет типа string
* Набор точек типа vector<Point>

В классе реализованы следующие методы:

* Конструктор
* Метод перемещения центра фигуры по координатам *void reLocate (double x, double y)*
* Метод для изменения размера фигуры *virtual void zoom(double k)*
* Вспомогательный метод для изменения размера *void prepare\_for\_zoom(double& k)*
* Метод для поворота на заданный угол, вокруг центральной точки *void turn(int deg)*
* Метод изменения цвета фигуры *void setColour(string Color)*
* Перегружен оператор вывода в поток *friend std::ostream& operator<<(std::ostream& out, Shape& sh)*
* Метод для вывода информации *virtual void printOUT(std::ostream& out)*

1. Написан класс Square, наследуемый от Shape.

Хранит переменные:

* Сторону квадрата side типа double

Методы класса:

* Переопределен конструктор
* Переопределен метод вывода
* Переопределен метод масштабирования

1. Написан класс Rhomb, наследуемый от Shape.

Хранит переменные:

* Cторона side типа double

Методы класса:

* Переопределен конструктор
* Переопределен метод вывода
* Переопределен метод масштабирования

1. Написан класс Trapezium, наследуемый от Shape.

Хранит переменные:

* Верхнее основание типа double
* Нижнее основание типа double
* Высоту типа double

Методы класса:

* Переопределен конструктор
* Переопределен метод вывода
* Переопределен метод масштабирования

1. Написаны тесты для классов Shape, Pentagon, Rectangle и Ellipse.
2. Код представлен в приложении А.
3. Нарисована UML-диаграмма классов. Рисунок в приложении Б.

**Выводы.**

Изучено наследование классов в C++. Написан базовый класс Shape, от которого наследуются Square, Rhomb и Trapezium.

Приложение А.

**КОД ПРОГРАММЫ LR2.CPP**

#include <iostream>

#include <string>

#include <vector>

#include <cmath>

#include <locale.h>

using namespace std;

#define PI 3.14159265358979323846

class Point {

public:

double x;

double y;

Point(double x, double y) : x(x), y(y) {}

};

class Shape {

public:

Shape(double x, double y) :color("white"), center(x, y), angle(0) {}

void setColor(string color) {// установка цвета

this->color = color;

}

void reLocate(double x, double y) {// перемещение в другие координаты (по центру)+

for (auto& i : points) {

i.x += x - center.x;

i.y += y - center.y;

}

center.x = x;

center.y = y;

}

virtual void zoom(double n) = 0;

void prepare\_for\_zoom(double& n) {// масштабирование каждой точки

if (n<0) {

n = abs(n);

}

center.x \*= n;

center.y \*= n;

for (auto& i : points) {

i.x \*= n;

i.y \*= n;

}

}

void turn(int angl) {// поворот на угол

angl = angl % 360;

angle += angl;

double radian = angl\*PI / 180;

for (auto& i : points) {

double x\_ = i.x\*cos(radian) - i.y\*sin(radian);

double y\_ = i.y\*cos(radian) + i.x\*sin(radian);

i.x = x\_;

i.y = y\_;

}

}

virtual void printOUT(ostream& out) = 0; // печать информации о фигуре

friend std::ostream& operator<<(std::ostream& outStream, Shape& shape) {

shape.printOUT(outStream);

return outStream;

}

protected:

string color;

Point center;

int angle;

vector<Point> points;

};

class Square : public Shape {

public:

Square(Point center, double side) : Shape(center.x, center.y), side(side) {

points.push\_back({ center.x - side / 2, center.y + side / 2 });

points.push\_back({ center.x + side / 2, center.y + side / 2 });

points.push\_back({ center.x + side / 2, center.y - side / 2 });

points.push\_back({ center.x - side / 2, center.y - side / 2 });

}

void zoom(double n) {

prepare\_for\_zoom(n);

side \*= n;

}

void printOUT(ostream& out) {

out << "Создан квадрат!" << endl;

out << "Координаты центра: (" << center.x << ", " << center.y << ")" << endl;

out << "Длина стороны квадрата: " << side << endl;

out << "Точки квадрата:" << endl;

for (const auto& i : points) {

out << "(" << i.x << ", " << i.y << ")\n";

}

out << "Цвет: " << color << endl;

out << "Угол: " << angle << endl;

}

private:

double side;

};

class Rhomb : public Shape {

public:

Rhomb(Point center, double diagonal\_half\_1, double diagonal\_half\_2) : Shape(center.x, center.y), side(sqrt(pow(diagonal\_half\_1, 2) + pow(diagonal\_half\_2, 2))) {

points.push\_back({ center.x - diagonal\_half\_1, center.y });

points.push\_back({ center.x, center.y + diagonal\_half\_2 });

points.push\_back({ center.x + diagonal\_half\_1, center.y });

points.push\_back({ center.x , center.y - diagonal\_half\_2 });

}

void zoom(double n) {

prepare\_for\_zoom(n);

side \*= n;

}

void printOUT(ostream& out) {

out << "Создан ромб!" << endl;

out << "Координаты центра: (" << center.x << ", " << center.y << ")" << endl;

out << "Длина стороны ромба: " << side << endl;

out << "Точки ромба:" << endl;

for (const auto& i : points) {

out << "(" << i.x << ", " << i.y << ")\n";

}

out << "Цвет: " << color << endl;

out << "Угол: " << angle << endl;

}

private:

double side;

};

class Trapezium : public Shape {

public:

Trapezium(Point center, double side\_1, double side\_2, double h) : Shape(center.x, center.y), side\_1(side\_1), side\_2(side\_2), h(h) {

points.push\_back({ center.x - side\_1/2, center.y+h/2 });

points.push\_back({ center.x + side\_1 / 2, center.y + h / 2 });

points.push\_back({ center.x + side\_2 / 2, center.y - h / 2 });

points.push\_back({ center.x - side\_2 / 2, center.y - h / 2 });

}

void zoom(double n) {

prepare\_for\_zoom(n);

side\_1 \*= n;

side\_2 \*= n;

h \*= n;

}

void printOUT(ostream& out) {

out << "Создана трапеция!" << endl;

out << "Координаты центра: (" << center.x << ", " << center.y << ")" << endl;

out << "Длина оснований трапеции: " << side\_1 <<" "<<side\_2 <<endl;

out << "Длина высоты: " << h << endl;

out << "Точки трапеции:" << endl;

for (const auto& i : points) {

out << "(" << i.x << ", " << i.y << ")\n";

}

out << "Цвет: " << color << endl;

out << "Угол: " << angle << endl;

}

private:

double side\_1;

double side\_2;

double h;

};

int main()

{

setlocale(LC\_ALL, "Russian");

cout << "/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/" << endl;

cout << "/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/" << endl;

Square sq({ 0,0 }, 2);

cout << sq << endl;

cout << "/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/" << endl;

sq.zoom(3);

sq.turn(180);

sq.setColor("Green");

cout << sq<<endl;

cout << "/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/" << endl;

sq.reLocate(5.5, 7.6);

cout << sq;

cout << "/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/" << endl;

cout << "/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/" << "\n\n";

cout << "/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/" << endl;

cout << "/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/" << endl;

Rhomb rm({ 0,0 }, 2, 3);

cout << rm;

cout << "/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/" << endl;

rm.zoom(3);

rm.turn(180);

rm.setColor("Red");

cout << rm << endl;

cout << "/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/" << endl;

rm.reLocate(-5.5, -4.6);

cout << rm;

cout << "/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/" << endl;

cout << "/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/" << "\n\n";

cout << "/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/" << endl;

cout << "/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/" << endl;

Trapezium tr({ 0,0 }, 2, 5, 4);

cout << tr;

cout << "/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/" << endl;

tr.zoom(3);

tr.turn(180);

tr.setColor("Yellow");

cout << tr << endl;

cout << "/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/" << endl;

tr.reLocate(-5.5, -4.6);

cout << tr;

cout << "/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/" << endl;

cout << "/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/" << "\n\n";

return 0;

}

Приложение B.

**КОД ПРОГРАММЫ LR2.CPP**

